ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

KATEDRA GEOMATIKY

Úloha č. 4: Množinové operace s polygony

Monika KŘÍŽOVÁ
Marek HOFFMANN

leden 2022

Obsah

1 Zadání **2**

2 Údaje o bonusových úlohách **4**

3 Popis problému **4**

4 Popisy algoritmů **4**
4.1 Výpočet průsečíků A, B, setřídění 5
4.2 Update A, B 5
4.3 Ohodnocení vrcholů A, B dle pozice vůči B, A 5
4.4 Výběr vrcholů dle ohodnocení 5
4.5 Vytvoření hran 6

5 Vstupní data **6**
5.1 Načítání textového souboru 6
5.2 Kreslení polygonů 7

6 Výstupní data **8**

7 Printscreen vytvořené aplikace **11**

8 Dokumentace **12**
8.1 Hlavičkový soubor Types 12
8.2 Třída Algorithms 13
8.3 Třída Draw 16
8.4 Třída Edge 17
8.5 Třída QPointFBO 17
8.6 Třída SortByX 18
8.7 Třída SortByY 18
8.8 Třída Widget 18

9 Závěr **19**

1 Zadání

Vytvořte aplikaci, jež bude načítat polygony z připraveného souboru do grafického uživatelského rozhraní a v níž budou implementovány množinové operace - průnik, sjednocení a rozdíly. Celý text zadání je vložen na následující stranu dokumentu.

Úloha č. 4: Množinové operace s polygony

Vstup: množina n polygonů P=$\{P_1, ..., P_n\}$.

Výstup: množina m polygonů $P' = \{P'_1, ..., P'_m\}$.

S využitím algoritmu pro množinové operace s polygony implementujte pro libovolné dva polygony $P_i, P_j \in P$ následující operace:

- Průnik polygonů $P_i \cap P_j$,
- Sjednocení polygonů $P_i \cup P_j$,
- Rozdíl polygonů: $P_i \cap \overline{P}_j$, resp. $P_j \cap \overline{P}_i$.

Jako vstupní data použijte existující kartografická data (např. konvertované shape fily) či syntetická data, která budou načítána z textového souboru ve Vámi zvoleném formátu.

Grafické rozhraní realizujte s využitím frameworku QT.

Při zpracování se snažte postihnout nejčastější singulární případy: společný vrchol, společná část segmentu, společný celý segment či více společných segmentů. Ošetřete situace, kdy výsledkem není 2D entita, ale 0D či 1D entita.

Pro výše uvedené účely je nutné mít řádně odladěny algoritmy z úlohy 1. Postup ošetření těchto případů diskutujte v technické zprávě, zamyslete se nad dalšími singularitami, které mohou nastat.

Hodnocení:

Krok	**Hodnocení**
Množinové operace: průnik, sjednocení, rozdíl	20b
Konstrukce offsetu (bufferu)	*+10b*
Výpočet průsečíků segmentů algoritmem Bentley & Ottman	*+8b*
Řešení pro polygony obsahující holes (otvory)	*+6b*
Max celkem:	**44b**

Čas zpracování: 2 týdny

2 Údaje o bonusových úlohách

V rámci řešení nebyly vypracovány žádné bonusové úlohy.

3 Popis problému

Množinové operace patří mezi základní algoritmy GIS. Existuje několik druhů množinových algoritmů, mezi základní můžeme zařadit sjednocení (*Union*), které vytváří ze dvou množin množinu novou, která obsahuje všechny prvkou obou vstupních množin. Další operací je průnik (*Intersection*), jež vytváří ze vstupních množin takovou množinu, která zahrnuje pouze prvky náležející oběma množinám. Posledními zpracovávanými operacemi jsou rozdíly (*Difference*), neboli operace, které obsahují dle zadání buď všechny prvky z první množiny, ale zároveň žádný prvek z druhé množiny, nebo naopak všechny prvky z druhé množiny, ale žádný z množiny první.

4 Popisy algoritmů

Algoritmy fungují na tomtéž principu, jediným rozdílem je výběr hran podle požadované operace. Množiny budou pro další popis popsány jako A a B. Algortimus zahrnuje následující fáze:

1. Výpočet průsečíků A, B, setřídění.
2. Update A, B.
3. Ohodnocení vrcholů A, B dle pozice vůči B, A.
4. Výběr vrcholů dle ohodnocení.
5. Vytvoření hran.
6. Spojení hran do oblastí.

Pro snadnější práci byl vytvořen nový datový typ *QPointFBO*, jenž obsahuje kromě souřadnic bodu i hodnoty α a β, které poskytují informaci o poloze bodu vůči polygonu.

4.1 Výpočet průsečíků *A, B*, setřídění

Pro výpočet průsečíků byla použita metoda *Line Sweep*. Segmenty polygonu označíme jako e_i a e_j, průsečíky jako b_{ij}. Prlsečíky vypočítáme dle následující rovnice:

$$b_{ij} = p_i + \alpha \overrightarrow{u} = p_j + \beta \overrightarrow{v}, \tag{1}$$

kde $\overrightarrow{u}$ a $\overrightarrow{v}$ jsou směrové vektory:

$$\overrightarrow{u} = (x_{i+1} - x_i, yi + 1 - y_i), \overrightarrow{v} = (x_{j+1} - x_j, yj + 1 - y_j) \tag{2}$$

a α, β určují polohu bodu vůči polygonu:

$$\alpha = \frac{k_1}{k_3}, \beta = \frac{k_2}{k_3}, k_1 = v_x w_y - v_y w_x, k_2 = u_x w_y - u_y w_x, k_3 = v_y u_x - v_x u_y \tag{3}$$

Je-li $k_1 = 0$, $k_2 = 0$ a $k_3 = 0$, je e_i a e_j kolineární, je-li $k_1 = 0$ a $k_2 = 0$, je e_i a e_j rovnoběžné $\alpha \epsilon$ <0, 1> a $\beta \epsilon$ <0, 1> $\exists$ b_{ij}, jinak mimoběžné.

4.2 Update *A, B*

S každým nalezeným průsečíkem musí být aktualizována množina *B*. Průsečík b_{ij} se vkládá mezi body p_j a p_{j+1}, množina *A* se aktualizuje hromadně až po nalezení všech průsečíků, během procesu hledání se ukládá do mapy $M(\alpha)$

4.3 Ohodnocení vrcholů *A, B* dle pozice vůči *B, A*

O poloze e_i = (p_i, p_{i+1}) k B rozhoduje poloha libovolného vnitřního bodu $\overline{p_i}$ = 0.5(p_i + p_{i+1}), analogicky pro e_j= (q_j, q_{j+1}) vzhledem k A $\overline{q_j}$ = 0.5(q_j + q_{j+1}). Každému p_i *A* určíme polohu vzhledem k *B*. Každému q_j *B* určíme polohu vzhledem k *A*.

Polohu přímky vůči polygonu ohodnotíme pomocí vlastního datového typu Enum, který zahrnuje hodnoty *Inner, Outer, Boundary*, tedy uvnitř, vně nebo na hranici polygonu.

4.4 Výběr vrcholů dle ohodnocení

Hrany jsou vybírány dle požadované množinové operace.

Při výběru *Průniku - Intersection* je nutné míti pro obě množiny hodnota *Outer*.

Pro tvorbu *Sjednocení - Union* se vybírá pro obě množiny hodnota *Outer*.

Pro tvorbu *Rozdílu - Difference B-A* se vybírá pro množinu *A* hodnota *Outer* a pro množinu *B* hodnota *Inner*.

Pro tvorbu *Rozdílu - Difference A-B* se naopak vybírá pro množinu *A* hodnota *Inner* a pro množinu *B* hodnota *Outer*.

4.5 Vytvoření hran

Ze všech hran mezi vrcholy a průsečíky množin se vyberou ty hrany, které mají požadované ohodnocení (dle vlastností výše) a vytvoří se nová množina vytvořená pouze těmito hranami.

5 Vstupní data

5.1 Načítání textového souboru

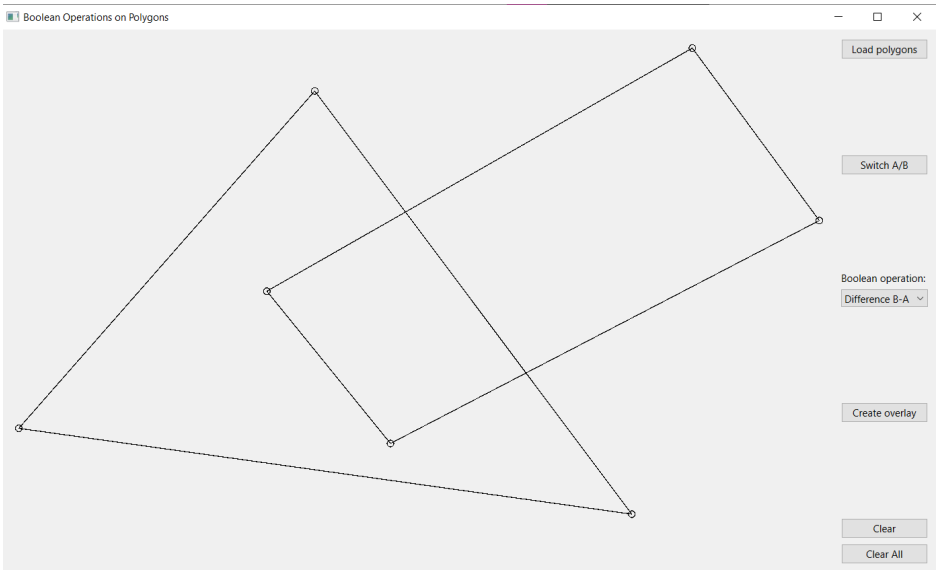
Soubor polygonů určenými body s polohovými souřadnicemi x, y v souřadnicovém systému S-JTSK se do projektu nahrávají po stisknutí tlačítka *Load polygons*.

Body jsou načítány postupně řádek po řádku, přičemž musí v nahrávaném souboru platit následující pravidla:

- na řádku je pořadí proměnných id » y (S-JTSK) » x (S-JTSK), hodnoty jsou od sebe odděleny jednou mezerou,
- id je identifikátor jednotlivých vrcholů polygonu, x, y, jsou polohové souřadnice vrcholu polygonu

```
1 743001.84 1039322.30
2 743003.27 1039321.21
3 743004.13 1039322.31
4 743003.91 1039322.48
5 743005.24 1039324.19
6 743007.20 1039326.72
7 743010.25 1039324.34
8 743010.50 1039324.36
9 743010.53 1039324.11
10 743013.48 1039321.84
```

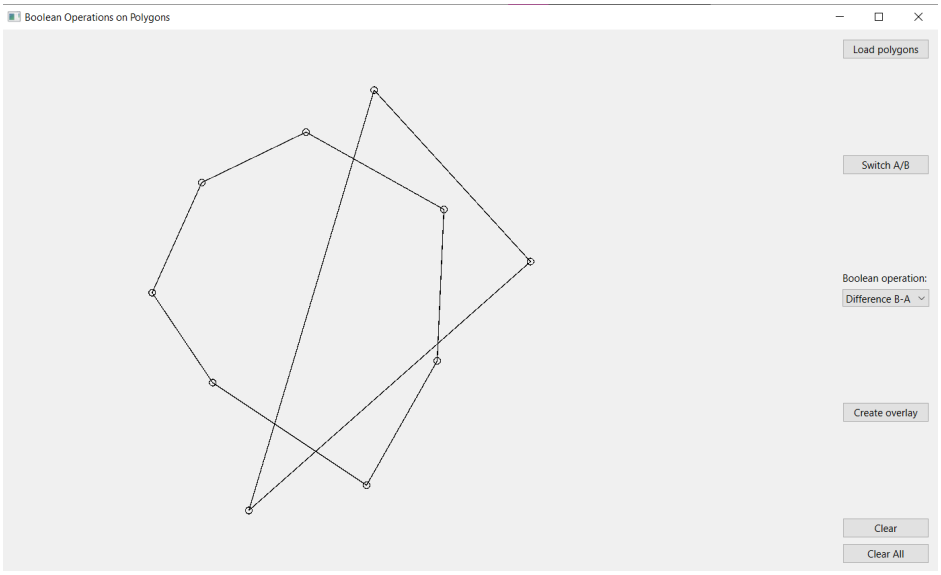
Obrázek 1: Špagetový model vstupního souboru bodů



Obrázek 2: Aplikace po načtení polygonů z textového souboru

5.2 Kreslení polygonů

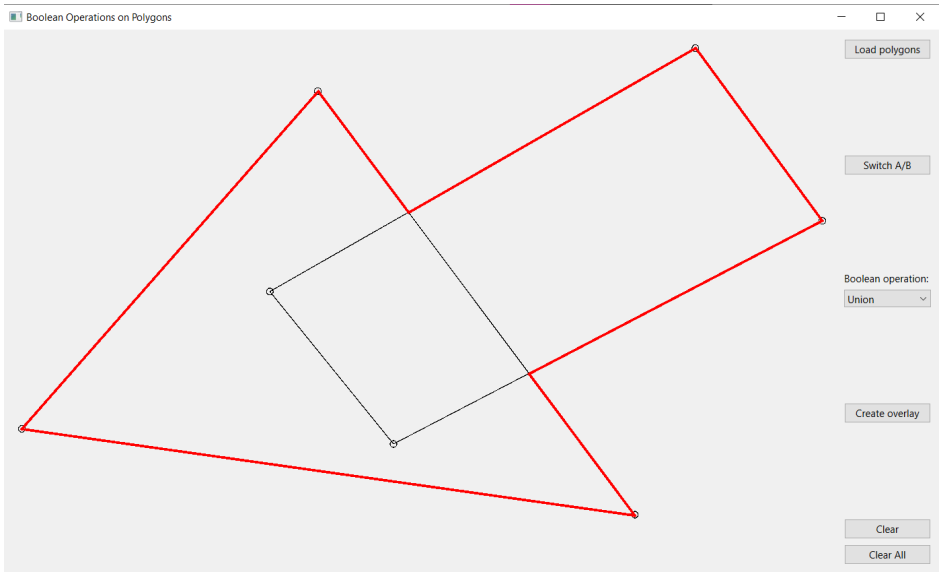
Druhou možností zadávání polygonů je kreslení polygonů přímo v interaktivním prostředí aplikace. Opakovaným zadáváním bodů kliknutím myši lze vytvořit polygon jakéhokoliv tvaru. Po stisknutí tlačítka *Switch A/B* lze stejným způsobem zadat druhý polygon. Nad touto dvojící pak lze provádět vybrané množinové operace.



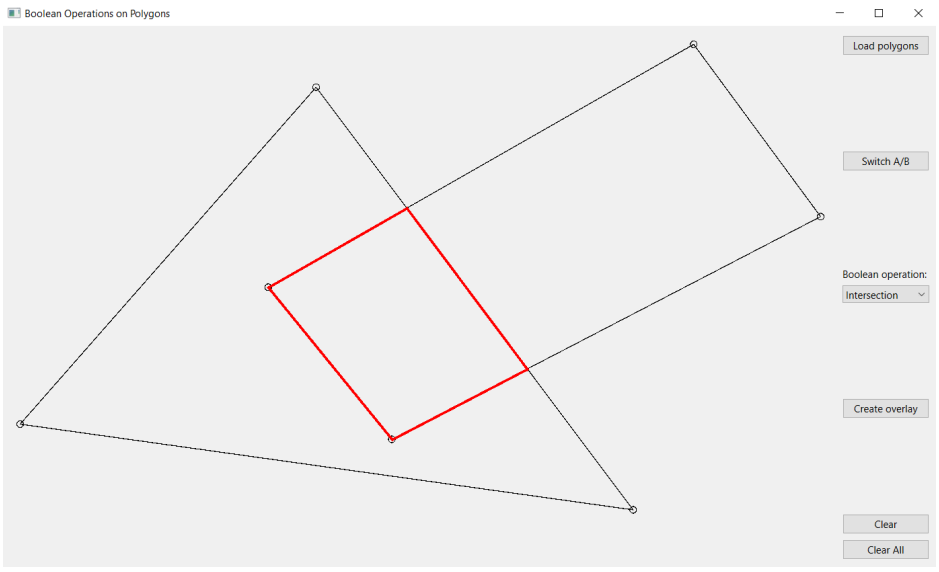
Obrázek 3: Aplikace po manuálním nakreslení dvou polygonů

6 Výstupní data

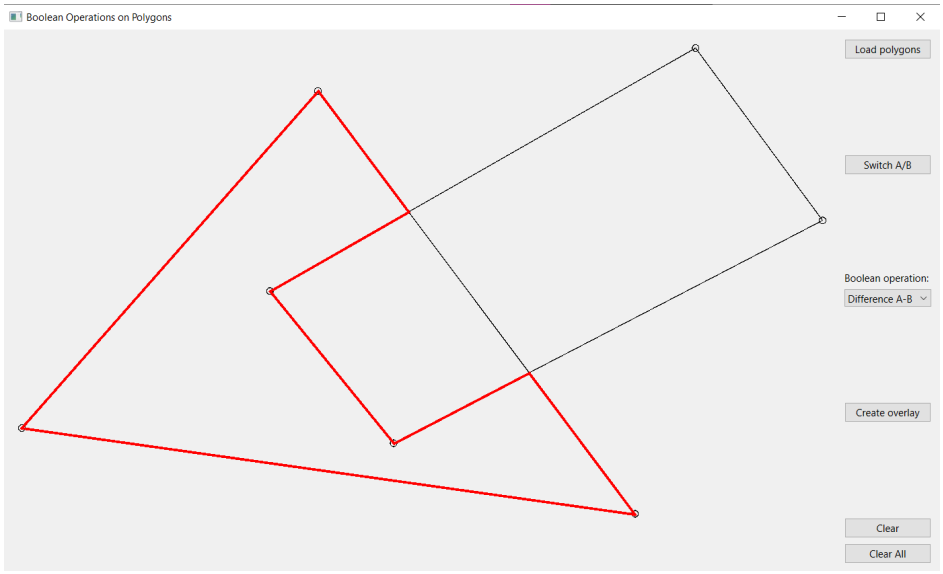
Nad zadanými polygony lze na základě výběru z comboBoxu provést 4 základní množinové operace; sjednocení (4), průnik (5) a rozdíl polygonů A-B a B-A (6, 7). Jednotlivé operace jsou zobrazeny na následujících obrázcích.



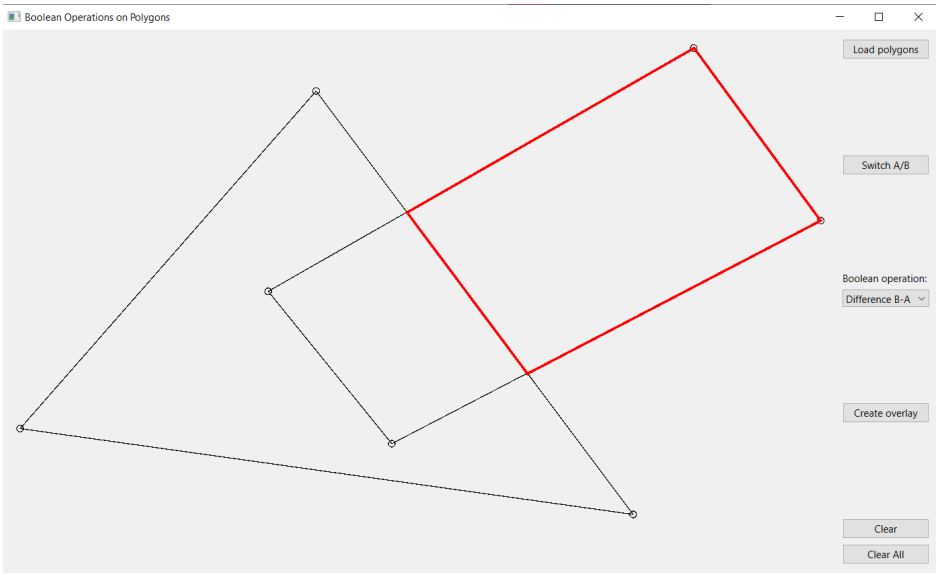
Obrázek 4: Sjednocení polygonů



Obrázek 5: Průník polygonů

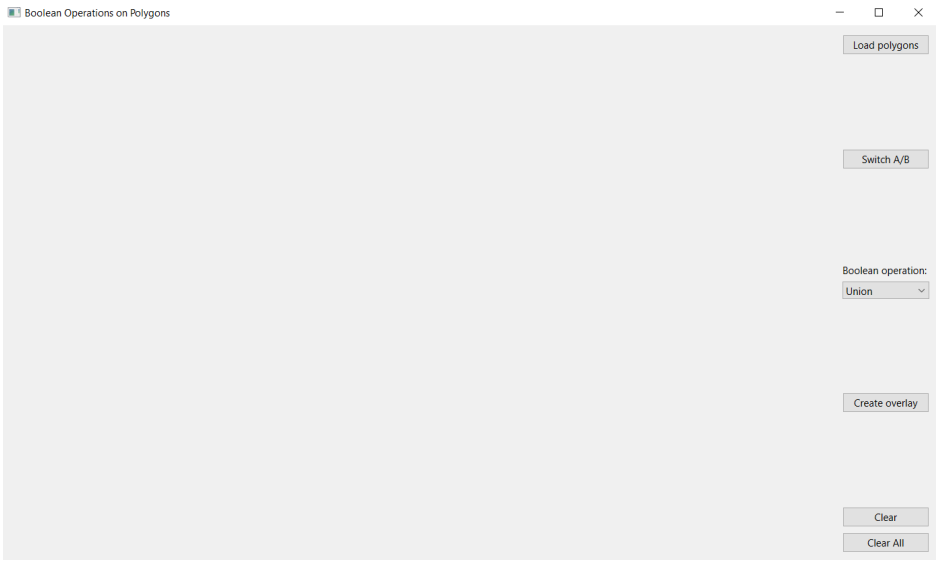


Obrázek 6: Rozdíl polygonů A-B



Obrázek 7: Rozdíl polygonů B-A

7 Printscreen vytvořené aplikace



Obrázek 8: Úvodní okno aplikace

8 Dokumentace

Kód zahrnuje celkem 7 tříd – Algorithms, Draw, Edge, QpointFBO, SortByX, SortByY Widget a jeden hlavičkový soubor Types. Všechny třídy budou následně detailněji popsány.

8.1 Hlavičkový soubor Types

Hlavičkový soubor Types definuje následující výčtové typy.

- TPointLinePosition
- TPointPolygonPosition
- TBooleanOperation
- T2LinesPosition
- TPolygon
- TEdges

TPointLinePosition TPointLinePosition ukládá hodnoty polohy bodu vůči přímce.

- RightHP v případě, že bod leží v pravé polorovině,
- LeftHP v případě, že bod leží v levé polorovině,
- On v případě, že bod leží na přímce.

TPointPolygonPosition TPointPolygonPosition ukládá hodnoty polohy bodu vůči polygonu.

- Inner v případě, že bod uvnitř polygonu
- Outer v případě, že bod leží vně polygonu
- Boundary v případě, že bod leží na hraně polygonu

TBooleanOperation TBooleanOperation definuje hodnoty pro určení typu množinové operace.

- Union - sjednocení polygonů
- Intersection - průnik polygonů
- DifferenceA_B - rozdíl polygonů A-B
- DifferenceB_A - rozdíl polygonů B-A

T2LinesPosition TPointPolygonPosition ukládá hodnoty pro určení vzájemné polohy dvou přímek.

- Colinear - kolineární přímky
- Parallel - rovnoběžné přímky
- Intersect - různoběžné přímky
- NonIntersect - mimoběžné přímky

TPolygon Představuje vektor datového typu QPointFBO.

TEdges Představuje vektor datového typu Edge

8.2 Třída Algorithms

Třída Algorithms obsahuje 9 funkcí:

- TPointLinePosition getPointLinePosition(QPointFBO &a, QPointFBO &p1, QPointFBO &p2);
- double get2LinesAngle(QPointFBO &p1, QPointFBO &p2, QPointFBO &p3, QPointFBO &p4);
- TPointPolygonPosition getPositionWinding(QPointFBO &q, TPolygon &pol);
- std::tuple<QPointFBO,T2LinesPosition> get2LinesIntersection(QPointFBO &p1, QPointFBO &p2, QPointFBO &p3, QPointFBO &p4);
- void updatePolygons(TPolygon &A, TPolygon &B);

- void processIntersection(QPointFBO &b, double t, int &index, TPolygon &P);
- void setEdgePositions(TPolygon &A, TPolygon &B);
- void selectEdges(TPolygon &P, TPointPolygonPosition pos, TEdges &edges);
- TEdges createOverlay(TPolygon &A, TPolygon &B, TBooleanOperation &op);

TPointLinePosition getPointLinePosition(QPointFBO &a, QPointFBO &p1, QPointFBO &p2); Analyzuje vzájemnou polohu mezi bodem a linií polygonu, resp. v jaké polorovině vůči linii se bod nachází. Vstupními argumenty funkce jsou souřadnice určovaného bodu (jako QPointFBO) a souřadnice 2 bodů určujících polohu linie (vrcholy polygonu taktéž jako QPointFBO). Funkce vrací vždy hodnotu RightHP, LeftHP, On datového typu TPointLinePosition dle následujících pravidel:

- RightHP v případě, že bod leží v pravé polorovině,
- LeftHP v případě, že bod leží v levé polorovině,
- On v případě, že bod leží na linii.

double get2LinesAngle(QPointFBO &p1, QPointFBO &p2, QPointFBO &p3, QPointFBO &p4); Počítá úhel mezi dvěma liniemi. Vstupními argumenty jsou body určující linie, tzn. vrcholy polygonu. Návratovou hodnotou funkce je double – desetinné číslo s velikostí úhlu mezi těmito přímkami.

TPointPolygonPosition getPositionWinding(QPointFBO &q, TPolygon &pol); Funkce analyzuje polohu boduvůči polygonu metodou Winding Number, jež byla podrobně vysvětlena v kapitole 3. Vstupními argumenty je bod typu QPointFB a polygon datového typu TPolygon. Výstupní hodnotou je datový typ TPointPolygonPosition, jehož hodnoty nabývají následující hodnoty:

- Inner v případě, že bod uvnitř polygonu
- Outer v případě, že bod leží vně polygonu
- Boundary v případě, že bod leží na hraně polygonu

std::tuple<QPointFBO,T2LinesPosition> get2LinesIntersection(QPointFBO &p1, QPointFBO &p2, QPointFBO &p3, QPointFBO &p4); Funkce určuje vzájemnou polohu dvou přímek na základně hodnot parametrů k1, k2 a k3, které se vypočetly jako dílčí determinanty směrových vektorů u,v,w. Pokud se všechny parametry k blíží k nule, vrací funkce dvojici hodnot QPointFBO s defaultními parametry a hodnotu **Colinear** datového typu T2LinesPosition. Pokud se blíží nule pouze koeficienty k1 a k2, vrací funkce QPointFBO s defaultními parametry a **Parallel**. Dále se vypočtou hodnoty α a β. Pokud se hodnoty nacházejí v intervalu (0,1), vypočte se pozice průsečíku. Následně funkce vrací hodnotu **Intersect** a bod QPointFBO s vypočtenými souřadnicemi a parametry α a β. Pokud nenastane ani jedna z výše popsaných situací, vrací funkce bod QPointFBO a hodnotu **NonIntersect** dat. typu T2LinesPosition.

void updatePolygons(TPolygon &A, TPolygon &B); Funkce hledá průsečíky dvou polygonů pomocí funkce *get2LinesIntersection()*. Pokud průsečík existuje, uloží se do mapy, jejíž klíčem je parametr α nebo β a hodnotou je vypočtený průsečík. Následně se bod přidá do daného polygonu pomocí funkce *processIntersection*. Funkce nic nevrací.

void processIntersection(QPointFBO &b, double t, int &index, TPolygon &P); Funkce je volána funkcí *updatePolygons()*. Tato funkce vkládá nalezený průsečík s hodnotami α nebo β na určitou pozici danou indexem v daném polygonu. Funkce nic nevrací.

void setEdgePositions(TPolygon &A, TPolygon &B); Funkce určuje pozici středových bodů všech hran polygonu vůči druhému polygonu. K určení vzájemné polohy body a polygonu se volá funkce *getPositionWinding()*. Settrem se každému vrcholu polygonu nastaví hodnota TPointPolygonPosition. Funkce nic nevrací.

void selectEdges(TPolygon &P, TPointPolygonPosition pos, TEdges &edges); Funkce přidává do vektoru hran TEdges ty hrany polygonu P, jejichž počáteční body mají hodnotu TPointPolygonPosition shodnou se vstupní hodnotou pos. Funkce nic nevrací.

TEdges createOverlay(TPolygon &A, TPolygon &B, TBooleanOperation &op); Funkce vrací ty hrany obou polygonů, které odpovídají vstupní hodnotě TBooleanOperation, která odpovídá zvolené množinové operaci.

Funkce nejdříve vypočte průsečíky polygonů a aktualizuje je pomocí funkce *updatePolygons()*. Následně se určí vzájemnou polohu hran vůči oboum polygonům pomocí funkce *setEdgePositions()*. Následně se opakovaně volá funkce selectEdges na základně vybrané množinové operace. Pokud je zvolena operace **Union** (sjednocení), vyberou se vnější hrany obou polygonů. Pokud se zvolí operace **Intersection** (průnik), za výsledek se uvažují pouze vnitřní hrany obou polygonů. Pokud je vybrána operace **Difference A-B**, vrátí funkce vnější hrany polygonu A a vnitřní hrany polygonu B. Pokud zvolíme operaci **Difference B-A**, vyberou se naopak vnitřní hrany hrany polygonu A a vnější hrany polygonu B.

8.3 Třída Draw

Třída Draw obsahuje následující funkce, které budou dále podrobně popsány:

- void paintEvent(QPaintEvent *event);
- void mousePressEvent(QMouseEvent *event);
- void switchSource()addA = !addA;
- void drawPolygon(TPolygon &pol, QPainter &qp);
- TPolygon getA()return A;
- TPolygon getB()return B;
- void setEdges(TEdges &edg)res = edg;
- void clear()res.clear();
- void clearAll()A.clear(); B.clear(); res.clear();
- void loadData(QString &file_name);

a následující proměnné:

- TPolygon A, B;
- TEdges res;
- bool addA;
- double y_max = 0, x_min = 999999999; //pro transformaci
- double y_min = 999999999, x_max = 0; //pro meritko

void paintEvent(QPaintEvent *event); Funkce nastavuje grafické atributy vykreslovaných objektů a kreslí na plátno zadané polygony a znázorňuje výsledky množinových operací.

void mousePressEvent(QMouseEvent *event); Metoda získává souřadnice kursoru myši, ukládá je do polygonu a následně je vykresluje na plátno.

void switchSource(); Metoda je volána po kliknutí na tlačítka **Switch A/B** a přepíná aktivní polygon.

void drawPolygon(TPolygon &pol, QPainter &qp); Metoda transformuje polygon na Qpolygon a vykresluje ho na plátno.

TPolygon getA(); Funkce pro vrácení polygonu A ze třídy Draw.

TPolygon getB(); Funkce pro vrácení polygonu A ze třídy Draw.

void setEdges(TEdges &edg)res = edg; Metoda ukládá vybrané hrany zadanou množinovou operací ze třídy Widget do třídy Draw.

void clear(); Metoda slouží pro odstranění výsledku množinové operace.

void clearAll(); Funkce odstraňuje obsah vykreslovacího plátna.

void loadData(QString &file_name); Funkce načítá data z textového souboru a ukládá je do polygonů A a B. Vstupní argumentem je cesta k souboru, jejž chceme načíst do aplikace.

8.4 Třída Edge

Ve třídě Edge je definován datový typ Edge, jenž definuje hranu polygonu. Hrana je definována počátečním a koncovým bodem uloženými v datovém typu QPointFBO. Funkce je opatřena gettry a settry.

8.5 Třída QPointFBO

Třída definuje datový typ QPointFBO, jež je dědičným datovým typem QPointF. Třída obsahuje soukromé proměnné alfa beta a pos, která ukládá pozici bodu vůči polygonu. Funkce je opatřena sadou gettrů a settrů.

8.6 Třída SortByX

Třída definuje operátor přetížení, který řadí body podle x-ové souřadnice.

8.7 Třída SortByY

Třída definuje operátor přetížení, který řadí body podle y-ové souřadnice.

8.8 Třída Widget

Třída Widget obsahuje 13 metod:

- void on_pushButton_clicked();
- void on_pushButton_2_clicked();
- void on_pushButton_3_clicked();
- void on_pushButton_4_clicked();
- void on_pushButton_5_clicked();

on_pushButton_clicked(); Funkce po stisknutí tlačítka **Switch A/B** volá funkci *switchSource()* ze třídy *Draw*

on_pushButton_2_clicked(); Funkce po stisknutí tlačítka **Create overlay** volá na základě zvolené množinové operace v *comboBoxu* metodu *createOverlay()* ze třídy *Algorithms* a vyhovující hrany ukládá setterem do třídy *Draw*.

on_pushButton_3_clicked(); Funkce po stisknutí tlačítka **Clear** volá metodu *clear()* ze třídy *Draw*.

on_pushButton_4_clicked(); Funkce po stisknutí tlačítka **Clear All** volá metodu *clearAll()* ze třídy *Draw*.

on_pushButton_5_clicked(); Funkce po stisknutí tlačítka **Load polygons** volá metodu *loadData()* ze třídy *Draw* a tím načítá soubor vybraný přes dialogové okno.

9 Závěr

V rámci úlohy byla vytvořena aplikace s grafickým rozhraním, která umožňuje provádět základní množinové operace nad dvěma polygony. Polygony lze do aplikace načíst dvěma způsoby. První možnost zahrnuje načítání z textového souboru. Aby bylo zajištěno správné načítání, musí soubor splňovat specifický datový formát. Alternativou je zadávat polygony manuálně. V tomto případě musí uživatel naklikat kursorem myší alespoň tři body u prvního polygonu a tlačítkem *Switch A/B* se přepnout do druhého polygonu a opět definovat jeho tvar. Abychom však mohli operace provádět, musí být zajištěno, že se oba polygony překrývají. Po načtení souboru pak lze provádět překryvné operace typu sjednocení, průnik a rozdíly polygonu A-B či B-A. Výsledek se v aplikaci zobrazí zvýrazněním hran tlustou červenou barvou. Výsledek operací lze pak příslušným tlačítkem odstranit, případně lze vyčistit celé plátno.